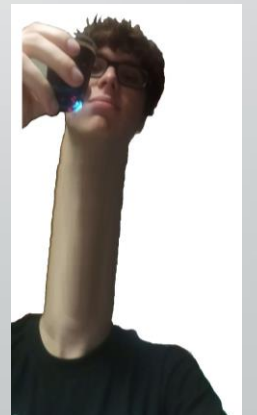


Elementi di Progettazione Software


Progetto Biblioteca - a.a. 2023/24



Corazzina Marco – Foccoli Matteo – Savoldi Tommaso

Inizializzazione

Log in Frame



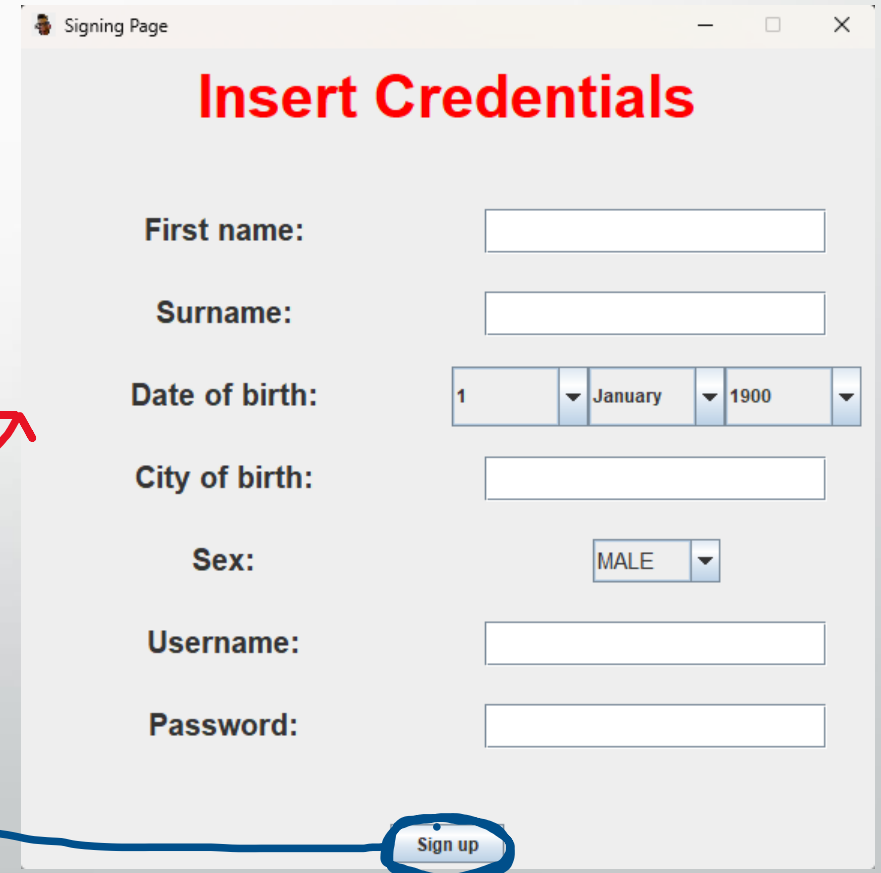
Library - Login

 SpiDo GUI Libreria 

Username:

Password:

Sign up Frame



Signing Page

Insert Credentials

First name:

Surname:

Date of birth:

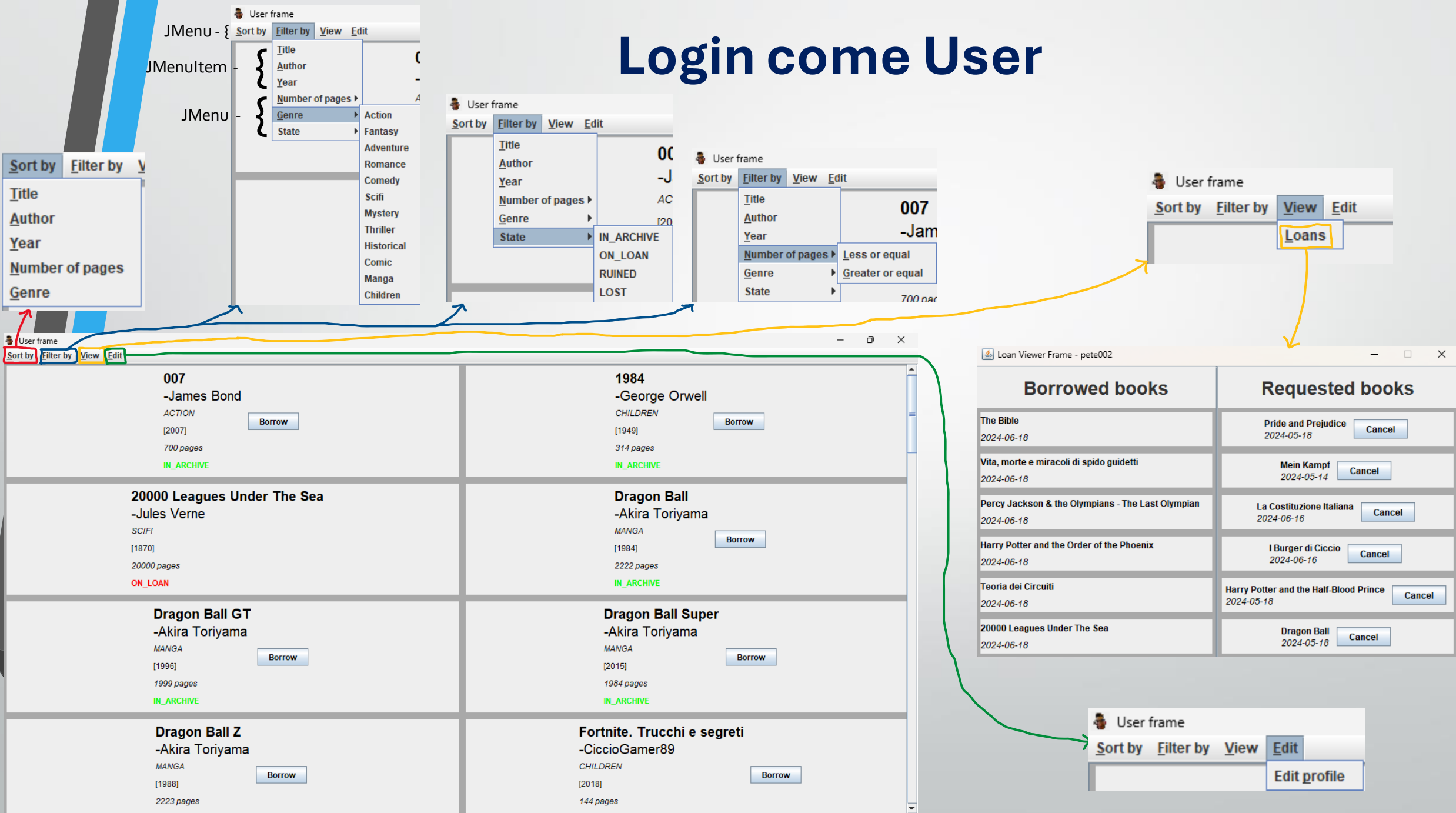
City of birth:

Sex:

Username:

Password:

Login come User



Login come Admin

- Edit

Admin - cyber

Sort by Filter by Edit View

Attack on Titan Vol.1 -Hajime Isayama <i>MANGA</i> [2012] 208 pages IN_ARCHIVE	Bibl -Jes <i>FANTA</i> [0] 7 page LOST
Elmer l'Elefante Variopinto -David McKee <i>CHILDREN</i> [1968] 16 pages	Fortnite. Truc -CiccioGamer8 <i>CHILDREN</i> [2018] 144 pages

Edit book

Insert book details

Title:

Author:

Release year:

Number of pages:

Genre:

Edit

Admin - cyber

Sort by Filter by Edit View

Add book

Add admin

Edit profile

Insert Credentials

First name:

Surname:

Date of birth:

City of birth:

Sex:

Username:
(can not be modified)

Password:
(blank = no update)

Edit

Add new book

Insert book details

Title:

Author:

Release year:

Number of pages:

Genre:

Add

Add new admin - cyber

Insert Credentials

First name:

Surname:

Date of birth:

City of birth:

Sex:

Username:

Password:

Add

Login come Admin

- View

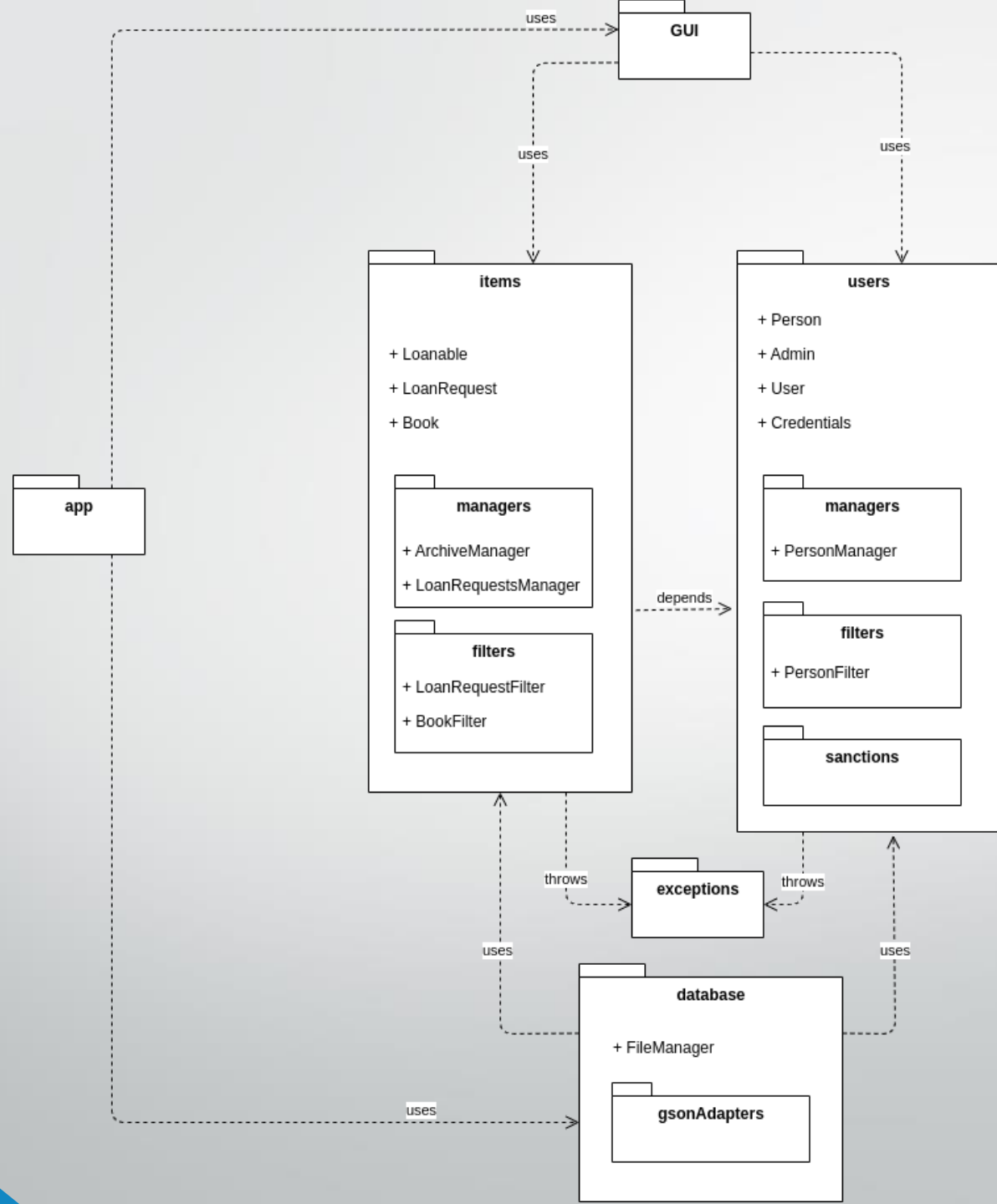
Admin - cyber	
Sort by	Filter by Edit View
Attack on Titan Vol.1 -Hajime Isayama MANGA [2012] 208 pages IN_ARCHIVE	Bibl -Jes FANTA [0] 7 page LOST
Elmer l'Elefante Variopinto -David McKee CHILDREN [1968] 16 pages	Fortnite. Truc -CiccioGamer8 CHILDREN [2018] 144 pages

pete002 2024-07-27	Vita, morte e miracoli di Spido Guidetti	Return
q 2024-06-18	I Burger di Ciccio	Return

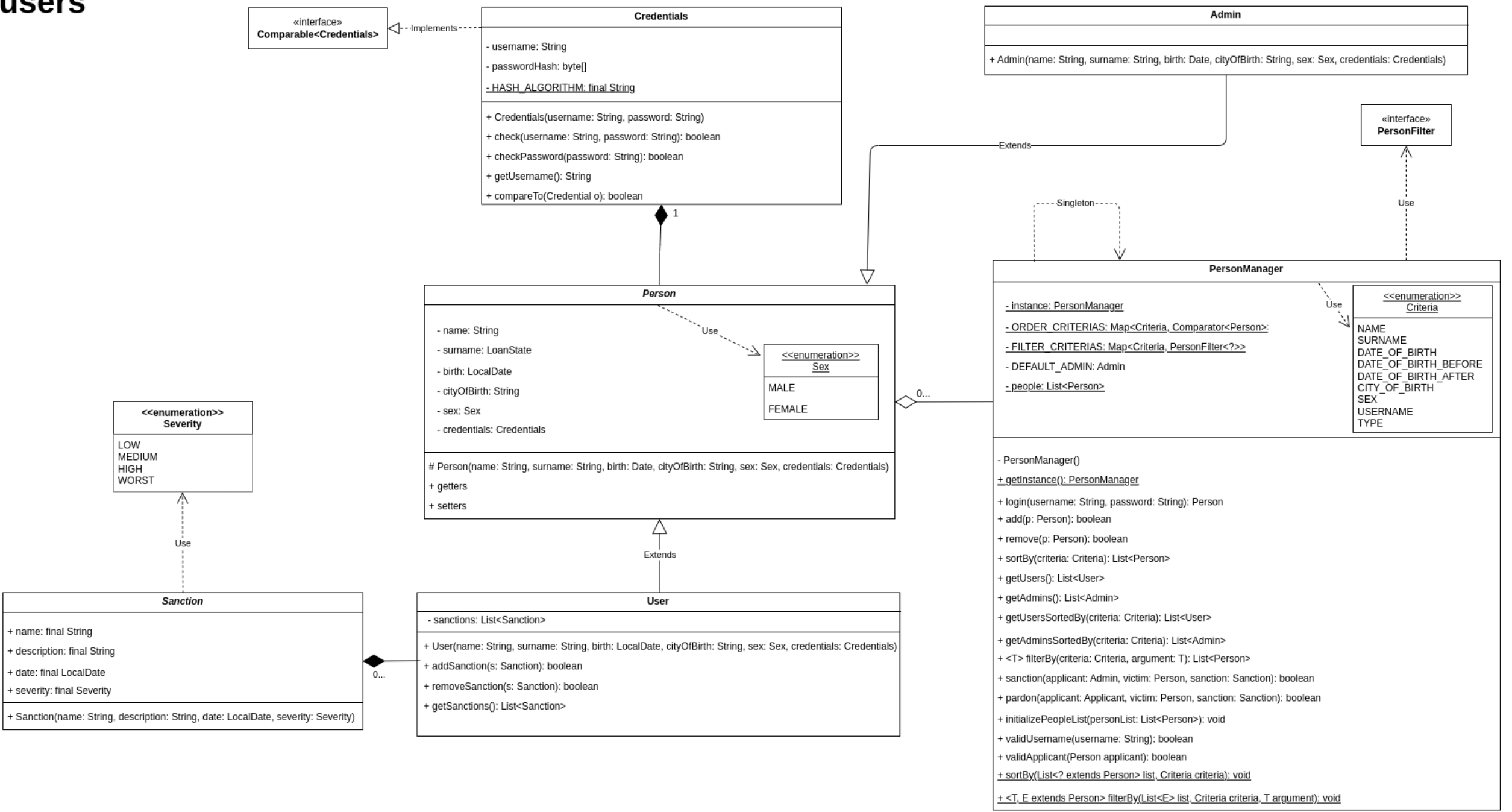
Sort by	Filter by	Edit	View
			View loans
			View requests
			View accounts

j 2024-06-14	Teoria dei Circuiti	Accept	Deny
phoenix 2024-05-18	V per Vendetta	Accept	Deny
q 2024-05-18	V per Vendetta	Accept	Deny
q 2024-05-18	Harry Potter e la Risposta in Frequenza	Accept	Deny

[U] q	Remove	Show info	Show loans	Show sanctions
[U] j	Remove	Show info	Show loans	Show sanctions
[U] pete002	Remove	Show info	Show loans	Show sanctions
[A] cyber	Remove	Show info		
[A] sudo	Remove	Show info		
[U] phoenix	Remove	Show info	Show loans	Show sanctions
[U] kibo	Remove	Show info	Show loans	Show sanctions



users



items



Il package users

Questo package offre la rappresentazione degli utenti che usufruiscono della biblioteca (che differiscono tra Admin e User), gestisce le loro credenziali di accesso e tramite il package “item” permette di richiedere libri in prestito.

Contiene diversi subpackage:

- **managers**: classi che permettono la gestione di utenti multipli
- **filters**: classi che aiutano i manager nell'atto del filtraggio della lista principale
- **sanctions**: racchiude le sanzioni che possono essere assegnate agli utenti

La classe Person e le sue sottoclassi Admin e User

La classe Person è la classe principale del package, contiene tutte le informazioni riguardo alla persona che sta usufruendo del servizio, quali:

- nome e cognome
- data e luogo di nascita
- sesso
- credenziali di accesso

Le sottoclassi Admin e User differenziano gli utenti in base ai privilegi. Gli User hanno una lista di sanzioni che gli sono state assegnate.

Credenziali di accesso - Credentials

La classe “Credentials” offre due campi esemplari:

- **String username** = username della persona
- **byte[] passwordHash** = l’hash della password della persona in byte array

Per non salvare la password come testo semplice (poco sicuro) è stato scelto di implementare l’hash con algoritmo SHA-256 ottenuto grazie alla classe di default MessageDigest.

Il controllo di accesso viene eseguito tramite il metodo *.check(username, password)* dove l’username viene confrontato col metodo *.compareTo()* della classe String mentre per la password ne viene calcolato l’hash e poi confrontato.

Gestione degli account - PersonManager

La classe PersonManager permette di gestire account multipli, recuperare l'account di login fornendo username e password, filtrare secondo varie condizioni ed effettuare il sorting della lista degli account.

Le possibilità di filtering e di sorting sono elencate nella enum Criteria e i predicati sono contenuti in due dizionari: FILTER_CRITERIAS e ORDERING_CRITERIAS. Il metodo di filtraggio o di riordino recupera autonomamente il predicato corretto.

Alcuni metodi (come la rimozione di una persona o l'aggiunta di una sanzione ad un utente) richiedono il passaggio di un oggetto Admin presente nella lista (che viene inteso come l'admin che sta richiedendo l'azione).

Punire gli utenti - Sanction

La classe astratta Sanction definisce la struttura di una sanzione che viene assegnata ad un utente quando esegue un'azione scorretta. Una sanzione infatti è composta da:

- **name** = nome (esplicativo e sintetico) della sanzione
- **description** = descrizione dettagliata della sanzione e del motivo per cui è stata assegnata
- **date** = data di assegnamento della sanzione
- **severity** = enum che indica la gravità della sanzione

La sottoclasse BookSanction (da cui ereditano tutte le altre) definisce una sanzione legata ad un libro in particolare e ne salva l'ID.

Il package “items”

Questo package offre le rappresentazioni di oggetti da poter dare in prestito, in questo caso libri, richieste di prestito e strutture dati adatte alla loro gestione.

Dipende dal package “users”.

Subpackages:

- **managers** - strutture dati adatte alla gestione dei libri della biblioteca e le richieste di prestito
- **filters** - interfacce funzionali usate per il filtraggio di libri e richieste nella visualizzazione

Loanable

- Classe astratta che rappresenta un qualsiasi oggetto che può essere dato in prestito.
- Ogni oggetto di questo tipo è caratterizzato da un ID univoco, un nome, uno stato del prestito, una data di riconsegna e la persona che ce l'ha in prestito.
- I termini del prestito, il nome e lo stato dell'oggetto possono essere cambiati solo da un admin
- Fornisce un enum i cui valori rappresentano lo stato dell'oggetto (in archivio, in prestito, perso, rovinato)
- Nella serializzazione l'unica informazione salvata dell'utente che ha in prestito l'oggetto è l'username

Book

- eredita da Loanable
- Rappresenta un libro della biblioteca che può essere dato in prestito
- Caratterizzato da titolo, autore, genere, anno di rilascio e numero di pagine
- Offre un enum di generi di libri
- Gli attributi possono essere modificati solo da un admin

LoanRequest

- Rappresenta una richiesta di prestito di un oggetto da parte di un utente
- Solo un User può fare una richiesta
- Solo un Admin può accettarla
- Caratterizzato dall'oggetto Loanable richiesto, dall'User richiedente, la data in cui la richiesta è stata fatta e lo stato di approvazione della richiesta
- E' possibile ottenere un oggetto in prestito solo facendo prima una richiesta, che poi deve essere accettata da un admin
- Nella serializzazione, l'unica informazione salvata dell'utente che ha in prestito l'oggetto è l'username, dell'oggetto richiesto viene solo salvato l'ID

ArchiveManager

- Fornisce la struttura dati per la gestione dei libri della biblioteca
- Singleton per garantire la presenza di una sola istanza all'interno del processo
- Una volta inizializzato, solo un admin può modificare i contenuti della biblioteca
- Fornisce metodi di ordinamento e filtraggio dei libri tramite mappe di Comparator e BookFilter e un enum che rappresenta all'esterno il criterio usato

LoanRequestsManager

- Fornisce la struttura dati per la gestione delle richieste di prestito
- Singleton per garantire la presenza di una sola istanza all'interno del processo
- Solo un Admin può accettare o rifiutare richieste
- Solo un User può effettuare una nuova richiesta; può cancellarne una propria
- Fornisce metodi di ordinamento e filtraggio delle richieste tramite mappe di Comparator e LoanRequestFilter e un enum che rappresenta all'esterno il criterio usato

Package “database” e Serializzazione

- I dati di utenti, libri, prestiti e richieste vengono salvati in 3 file JSON: accounts.json, archive.json e loanRequests.json nella cartella “assets”
- Per lo scopo viene usata la libreria open source GSON di Google
- FileManager contiene metodi statici che servono per serializzare e deserializzare i dati sopracitati